Supplementary Materials for DG-Recon: Depth-Guided Neural 3D Scene Reconstruction

Contents

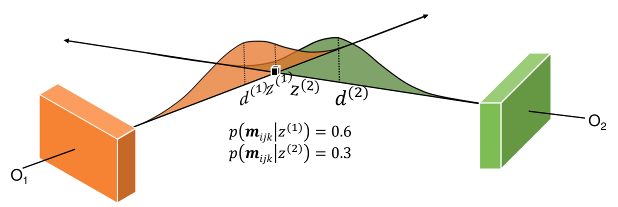
1. Occupancy mapping details

Assuming a static world, the occupancy mapping algorithm [15] divides the 3D space into voxels. The occupancy of a voxel m_{ijk} is a binary variable where $m_{ijk} = 1$ indicates the voxel is close to a surface and $m_{ijk} = 0$ otherwise. The probability of a voxel being occupied $p(m_{ijk})$ is approximated by the posterior probability given a stream of depth images $\mathbf{D}$ and its corresponding camera projection $\mathbf{P}$. A voxel is considered occupied if $p(m_{ijk}) > 0.5$, *i.e.*, the log-odds $l(m_{ijk}) > 0$. Algorithm 1 provides the pseudocode for the occupancy mapping algorithm.

Figure 1 illustrates the inverse sensor model for an example 3D point $\mathbf{x}_{ijk}$ observed from two different views. In one view, the 3D point is closer to the corresponding estimated depth, resulting in a higher probability of 0.6. In the other view, the 3D point is more distant from the estimated depth, resulting in a lower probability of 0.3. Given these two depth estimations, the integrated probability in log-odds notation $l(m_{ijk})$ is approximately -0.8 following Algorithm 1. The voxel is therefore considered not close to a surface given the information so far.

Algorithm 1 Occupancy Mapping Algorithm

```
for all voxels $ijk$ do
    $l(m_{ijk}) \leftarrow 0$                                  ▷ Initialization
end for
$\sigma \leftarrow 8$ voxel size
while receiving $\mathbf{D}^{(k)}, \mathbf{P}^{(k)}$ for keyframe $k$ do
    for all voxels $ijk$ do
        if voxel $ijk$ in frustrum of view $k$ then
            $u, v, z \leftarrow \mathbf{P}^{(k)}\mathbf{x}_{ijk}$              ▷ Projection
            $\mu = \mathbf{D}^{(k)}_{uv}$
            $p(m_{ijk}|z) \leftarrow \frac{\mathcal{N}(z|\mu,\sigma^2)}{\mathcal{N}(\mu|\mu,\sigma^2)}$
            $l(m_{ijk}) \leftarrow l(m_{ijk}) + \log \frac{p(m_{ijk}|z)}{1-p(m_{ijk}|z)}$
        end if
        $m_{ijk} \leftarrow l(m_{ijk}) > 0$                    ▷ Output Occupancy
    end for
end while
```



$d^{(1)}$
$z^{(1)}$
$z^{(2)}$
$d^{(2)}$
$p(\boldsymbol{m}_{ijk}|z^{(1)}) = 0.6$
$p(\boldsymbol{m}_{ijk}|z^{(2)}) = 0.3$
O_1
O_2

Figure 1. **Example of the occupancy probability model**. Given the distance to the camera center $z^{(k)}$ and the corresponding monocular depth estimation $d^{(k)}$, the occupancy probability of a voxel $p(m) = 1$ if $z^{(k)} = d^{(k)}$ and drops gradually to 0 when $z^{(k)}$ moves away from $d^{(k)}$. The probabilities from multiple views are integrated following the occupancy grid mapping (Algorithm 1) [15].

2. Reproducibility

Architectures. DG-Recon follows the same architecture as NeuralRecon [23] with a MnasNet [24] feature extractor and a SPVCNN [25] TSDF regressor which operates on 3 different scales with image feature sizes 120×160, 60×80, 30×40 and voxel sizes 4, 8, and 16cm respectively. The image feature dimensions are (24, 40, 80) for the three different scales and the volumetric feature dimensions are (28, 44, 84). Compared to the image features, the 4 additional dimensions in the volumetric features are for the geometry features.

The cross-attention-based fusion model is composed of 4 layers of Transformer decoders [26] without the self-attention layers. The number of heads for the multi-head attention is set to 4. The fusion also operates on three different scales with token dimensions (28, 44, 84) respectively from high resolution to low resolution. The dimensions of the feed forward layers are set to be (32, 64, 128) for the three corresponding scales. The initial query tokens are initialized with 1s and optimized through back-propagation.

The depth model consists of a ResNet34 [10] encoders and a convolutional decoder [19]. The decoder resembles features from four different scales of the ResNet34 backbone, varying from the conv1 block to the conv4 block. The number of decoder features is set to 128. The network predicts the inverse depth clipped between 0.1 and 10. A sparse inverse depth image, rendered from the sparse reconstructions, is concatenated to the RGB images as the inputs to the ResNet34 encoder. The pixels missing depth values (a vast majority in sparse depth input) are set to 0.

Hyperparameters. DG-Recon was trained 30 epochs using the Adam optimizer [11] on a single NVIDIA A100 with batch size 12. The learning rate was set to 1e-3 and was reduced by a factor of 2 every 12 epochs. The other training hyperparameters were kept the same as [23]. The depth model was trained with the Adam optimizer [11] for 30 epochs at a learning rate of 5e-5. The learning rate decays by 0.1 every 20 epochs. The training was executed on a single NVIDIA A100 GPU with batch size 16. The backbones (MnasNet and ResNet34) were initialized with ImageNet [4] pre-trained weights. The other weights were initialized randomly with the PyTorch [18] default uniform initializers, *i.e*. [13] for convolutional and feed-forward layers, and [9] for multi-head attention. All the inferences were executed on a single 11GB NVIDIA RTX 2080Ti GPU with batch size 1.

Visibility mask. Similar to [1, 22], visibility masks were applied during training and evaluating DG-Recon. The visibility mask was obtained by integrating the ground truth depth from all frames. A voxel is considered invisible if it has not been observed from any view, *i.e*., is out-of-frustum or is occluded. Occlusion is determined if the voxel is 12cm behind a surface indicated by the ground truth depths.

Marching Cubes. To obtain meshes from sparse TSDF volumes of DG-Recon, the marching cubes algorithm [14] was applied to extract surfaces at the zero level set. Note that the non-activated voxels in the sparse TSDF volumes are filled with 1s by default, indicating empty spaces. Naively running marching cubes would result in double-layer surfaces for walls, ceilings, and floors as also reported by [23]. Although the double-layer surfaces do not affect DG-Recon much in the evaluation because the second layers are often masked out by the visibility mask, it might be desirable to extract single-layer surfaces for many applications. That can be done in real-time by providing additional masks to marching cubes where the mask finds sharp transitions between -1 and 1. This method is built upon the observations that the TSDF values for desired surfaces transit smoothly around zeros whereas the TSDF values for undesired surfaces transit abruptly from -1 (activated voxels predicted to be behind the surface) to 1 (non-activated voxels). All the qualitative analyses were conducted with single-layer meshes extracted this way.

Sparse depth. The depth model of DG-Recon takes the sparse depths from the 6-degree-of-freedom localization pipeline as input. Because all three datasets did not dump the sparse depth images when running such pipelines [3, 17], we prepare the sparse depths ourselves using COLMAP [20]. More specifically, we feed the keyframes selected by [23] with the corresponding camera poses to COLMAP for keypoints extraction, keypoints matching, and point triangulation. The result sparse point clouds are then projected to the perspective view to get sparse depths for the selected keyframes. The sequential matching mode was used in this process and all the hyperparameters were kept default.

7-Scenes. Evaluation on 7-Scenes [8] is challenging, particularly because the camera and the depth sensor were not calibrated. We used the factory default calibrations for both sensors. The principle point was set to (320, 240) for both sensors and the focal length was set to (525, 525) for the camera and (585, 585) for the depth sensor. Because the Kinect camera has a different focal length from the camera used in ScanNet, the RGB images were resampled using the ScanNet camera intrinsic before feeding to DG-Recon and the other SOTA methods. Because the z-axis in the world coordinates of ScanNet points upwards, the world coordinates of 7-Scenes were rotated accordingly to match that. The ground truth meshes were reconstructed using the raw depth images, which were found empirically to be consistent with the official TSDF ground truth. Visibility masks were created and applied in the evaluation similar to the ScanNet experiments.

SUN3D. Evaluation on SUN3D [27] is in general similar to the evaluation of 7-Scenes. The RGB images were resampled and the world coordinates system was rotated to make the z-axis point upwards. The only difference is that the camera and the depth sensor were both calibrated with principle point (320, 240) and focal length (570, 570).

3. Metrics Definition

The definitions of the **3D reconstruction metrics** [16] are as follows.

- *Accuracy error (Acc)*: the average distance (cm) from the vertices p in the predicted mesh P to the closest vertex p^* in the ground truth mesh P^*

$$\frac{1}{N_P}\sum_{p \in P} \min_{p^* \in P^*} \|p - p^*\| \tag{1}$$

  where N_P is the number of vertices in the predicted mesh.

- *Completeness error (Comp)*: the average distance (cm) from the ground truth vertices to the closest predicted vertex

$$\frac{1}{N_{P^*}}\sum_{p^* \in P^*} \min_{p \in P} \|p - p^*\| \tag{2}$$

  where N_{P^*} is the number of vertices in the ground truth mesh.

- *Chamfer distance*: the average of the point-to-point distance (cm) from prediction to ground truth and from ground truth to prediction

$$\frac{1}{2}(\text{Acc} + \text{Comp}) \tag{3}$$

- *Precision (Prec)*: percentage of predicted vertices found ground-truth vertices within the radius of 5cm

$$\frac{1}{N_P}\sum_{p \in P}(\min_{p^* \in P^*} \|p - p^*\| < 5\text{cm}) \tag{4}$$

- *Recall (Rec)*: percentage of ground-truth points with corresponding predicted points within the radius of 5cm

$$\frac{1}{N_{P^*}}\sum_{p^* \in P^*}(\min_{p \in P} \|p - p^*\| < 5\text{cm}) \tag{5}$$

- *F-score*: the harmonic mean of precision and recall

$$\frac{2 \times \text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}} \tag{6}$$

The definitions of the **depth rendering metrics** [16] are as follows.

- *Abs Diff*: average of the absolute difference between the rendered depth d and the ground truth depth d^*

$$\frac{1}{n}\sum_{d \in \mathbf{D}} |d - d^*| \tag{7}$$

  where n is the number of valid pixels with $d > 0$ and $d^* > 0$

- *Abs Rel*: average of the absolute difference relative to the ground truth depth

$$\frac{1}{n}\sum_{d \in \mathbf{D}} |d - d^*|/d^* \tag{8}$$

- *Sq Rel*: average of the squared distance relative to the ground truth depth

$$\frac{1}{n}\sum_{d \in \mathbf{D}} |d - d^*|^2/d^* \tag{9}$$

- $\delta < 1.25$: the percentage of pixels with the maximum distance ratio smaller than 1.25

$$\frac{1}{n}\sum_{d \in \mathbf{D}}(\max(\frac{d}{d^*}, \frac{d^*}{d}) < 1.25) \tag{10}$$

- *Comp2D*: the percentage of valid depth from the rendered depth images

$$\frac{1}{N_D}\sum_{d \in \mathbf{D}}(d > 0) \tag{11}$$

  where N_D is the number of pixels.

4. More quantitative results

4.1. Comparison to parallel works

Table 1 compares DG-Recon and parallel works for reconstruction performance and runtime efficiency on ScanNet. Both CVRecon and FineRecon improve SOTA performance significantly for offline methods. Our best online model DG-Recon (var) is not any more on par with the best offline counterparts. But both our models still achieve the best accuracy-efficiency tradeoff among the online methods. DG-Recon (var) runs faster than VisFusion and achieve higher F-score and lower Chamfer distance. DG-Recon (c-att) runs slightly slower than VisFusion but reach remarkably better results in all columns. Both standard and high-resolution models from Zuo *et al*. run slower than DG-Recon and VisFusion. Even though not directly comparable to ours following protocol [1], their reconstruction performances were shown slightly lower than VisFusion following protocol [23] (0.572/0.589 vs. 0.598).

4.2. Varying the near-surface margin

Table 2 summarizes the reconstruction results and the volume sparsity for various choices of the near-surface margin, Δ. The volume sparsity for the $\Delta = 8$ model is 80%, 90%, and 96% for the three scales from coarse to fine. As the margin grow, the feature volume becomes denser. The $\Delta = 8$ model achieves the best reconstruction scores compared to the models trained with bigger and smaller near-surface margins.

Method	**Online**	**FPS↑**	**Acc. ↓**	**Comp.↓**	**Chamfer↓**	**Precision↑**	**Recall↑**	**F-score↑**
VoRTX[22]	x	2	4.31	7.23	5.77	0.767	0.651	0.703
CVRecon [6]	x	-	3.8	6.7	5.3	0.794	0.685	0.735
FineRecon [21]	x	-	5.25	5.11	5.18	0.780	0.734	0.755
VisFusion [7]	✓	25	4.17	9.05	6.61	0.751	0.580	0.653
Zuo *et al.* [28]	✓	13	-	-	-	-	-	-
Zuo *et al.* [28] (highres)	✓	5	-	-	-	-	-	-
DG-Recon (c-att)	✓	20	3.94	6.82	5.38	0.769	0.636	0.694
DG-Recon (var)	✓	34	4.40	6.49	5.44	0.732	0.628	0.674

Table 1. **Evaluation of the 3D mesh on the ScanNet test set**. Reconstruction results following the same evaluation protocol [1] for parallel works were taken from the corresponding papers. Frames per second (FPS) were reported by the original authors using similar GPU hardware, RTX 2080 Ti and RTX5000. Red cells mark the best number among the online methods, orange the second best, and yellow the third best. Mark - denotes not-yet-available measurement.

Δ	**Sparsity (%)↑**	**Acc ↓**	**Compl ↓**	**Prec ↑**	**Rec ↑**	**F-score ↑**
4	**90, 95, 98**	5.5	7.3	0.713	0.601	0.650
8	80, 90, 96	**5.4**	**7.1**	**0.718**	**0.613**	**0.659**
16	68, 80, 92	5.9	7.9	0.679	0.585	0.626

Table 2. **Effect of varying the near-surface margin**. Δ is expressed in the unit of voxel size. The margin shrinks and the sparsity increases when going from the coarse scale to the fine scale. The feature volume sparsity was measured at three different scales ranging from coarse to fine. The corresponding sparsity is reported accordingly from left to right.

The reconstruction results were measured on the ScanNet validation set. Different from the other ablation studies, single-layer meshes were produced following [23] and used in evaluation because the $\Delta = 4$ model sometimes creates double layers within the visibility mask.

4.3. Comparing self-attention and cross-attention

Fusion	**Sparsity** (%)	**Time** (ms) ↓	**GPU Memory** (MB) ↓
cross-attention	0	9.9	260
cross-attention	80	**3.4**	**60**
self-attention	0	31.5	490
self-attention	80	7.8	120

Table 3. **Comparison of cross-attention and self-attention**. Time and GPU memory were measured on the coarsest scale with volume size 32x32x32. The number of views was set to 9. The volume feature dimension was 84 and the feed-forward layer dimension was 128. 0% sparsity means the volume is fully occupied and 80% sparsity means 1/5 of the volume is occupied as measured using real case examples of ScanNet. All values were measured as an average of 100 repetitions.

Table 3 compares the cross-attention-based fusion against the self-attention-based fusion with and without sparse volume. The cross-attention-based fusion of sparse features is 10× and 2× faster respectively than the self-attention-based fusion with dense features [1] and with sparse features [22]. It consumes only 1/8 and 1/2 of the GPU memory compared to the dense and sparse self-attention-based modules. Even without sparsity, cross-attention-based fusion is 3× faster than self-attention-based fusion and occupies 50% of the GPU memory. The difference will be even bigger as the number of views grows. For example, [22] integrates features from 60 views instead 9 views.

Furthermore, Table 3 also shows that the sparsity introduced by depth-guided feature back projection is beneficial to the fusion efficiency. The runtime of cross-attention-based fusion drops from 9.9ms to 3.4ms and the GPU memory usage drops from 260MB to 60MB.

4.4. Varying distance threshold for evaluation

To expand the F-score evaluation, we also calculate F-scores at varying distance thresholds on 7Scenes [8]. Figure 2 shows that DG-Recon outperforms Atlas and NeuralRecon at all distance thresholds.

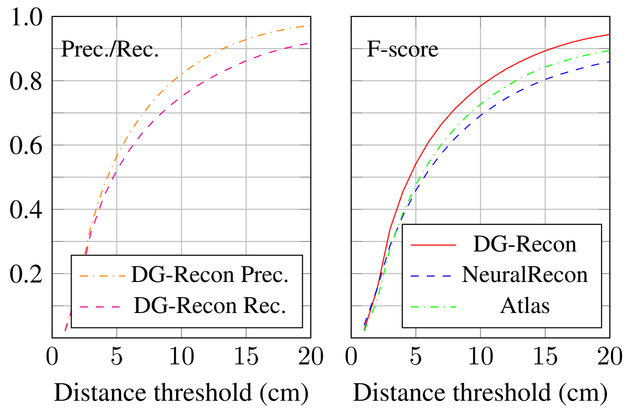


Figure 2. **Varying distance threshold on 7Scenes**. Note that the left figure differs from the classification Precision-Recall curve. The distance threshold does not control model prediction but determines true positives. Both precision and recall grow as distance threshold increases.

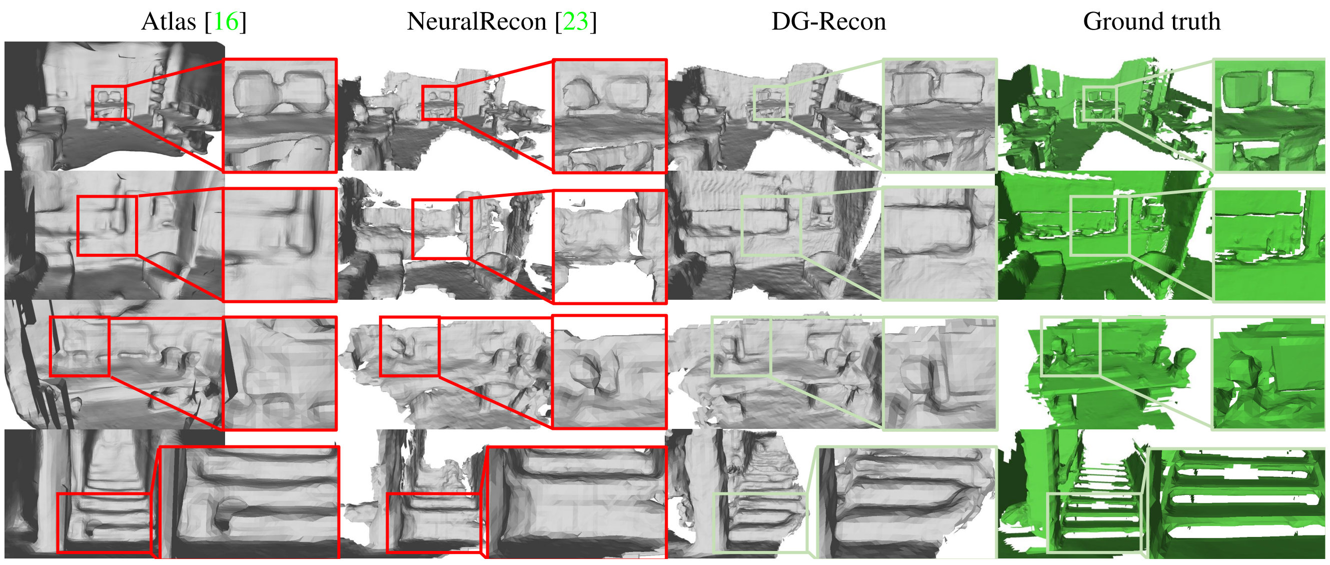


Figure 3. **Qualitative comparison of reconstructed meshes on 7-Scenes** [8]. Each row is a different scene. The red boxes mark over-smoothed or missing geometry while the light green boxes highlight the improved geometry by DG-Recon.

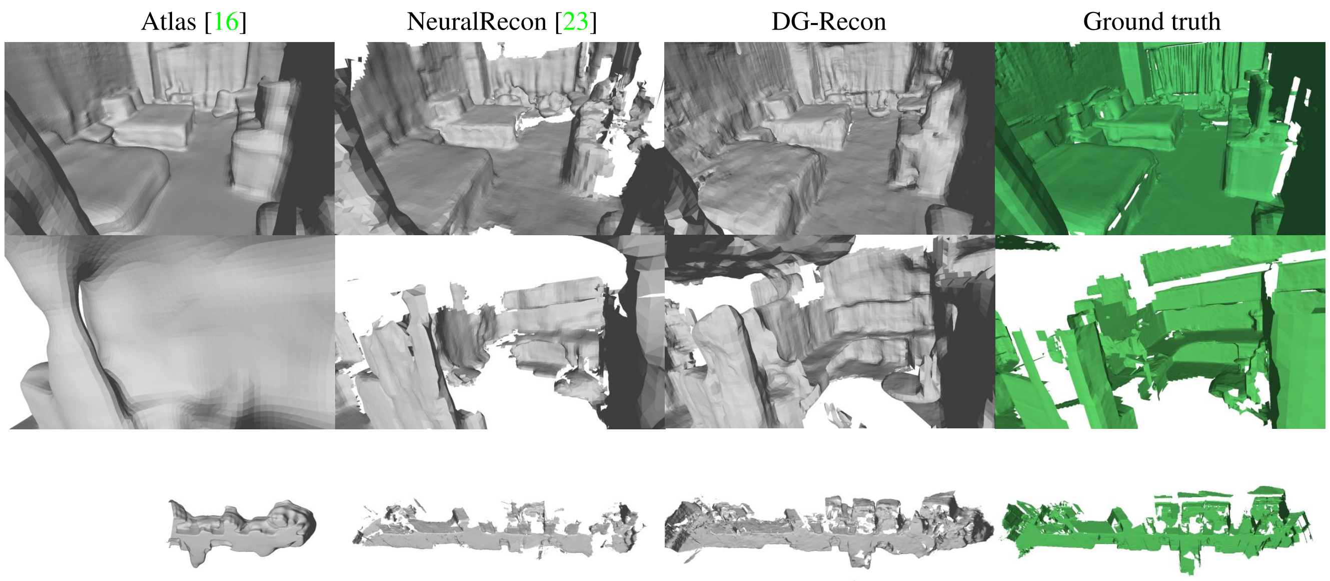


Figure 4. **Qualitative comparison of reconstructed meshes on SUN3D** [27]. The first row shows a hotel room. The third row gives an overview of the entire floor from Brown University. The second-row zoom-in to one of the office rooms on that floor.

5. More qualitative results

5.1. Qualitative analysis of 7-Scenes reconstruction

The qualitative comparison of the mesh reconstruction among Atlas [16], NeuralRecon [23] and DG-Recon on 7-Scenes [8] is illustrated in Figure 3. DG-Recon outputs geometry with sharper edges than the other two methods, *e.g.*, monitor, desk, and chairs in row 1, the kitchen worktops in row 2, three heads around the monitor in row 3, and stairs in row 4. The NeuralRecon baseline produces less complete reconstruction, for example, the cabinet in the kitchen is missing parts and the stairs are missing the first step.

5.2. Qualitative analysis of SUN3D reconstruction

Figure 4 illustrates how DG-Recon generalizes to the SUN3D [27] dataset, compared to Atlas [16] and NeuralRe-

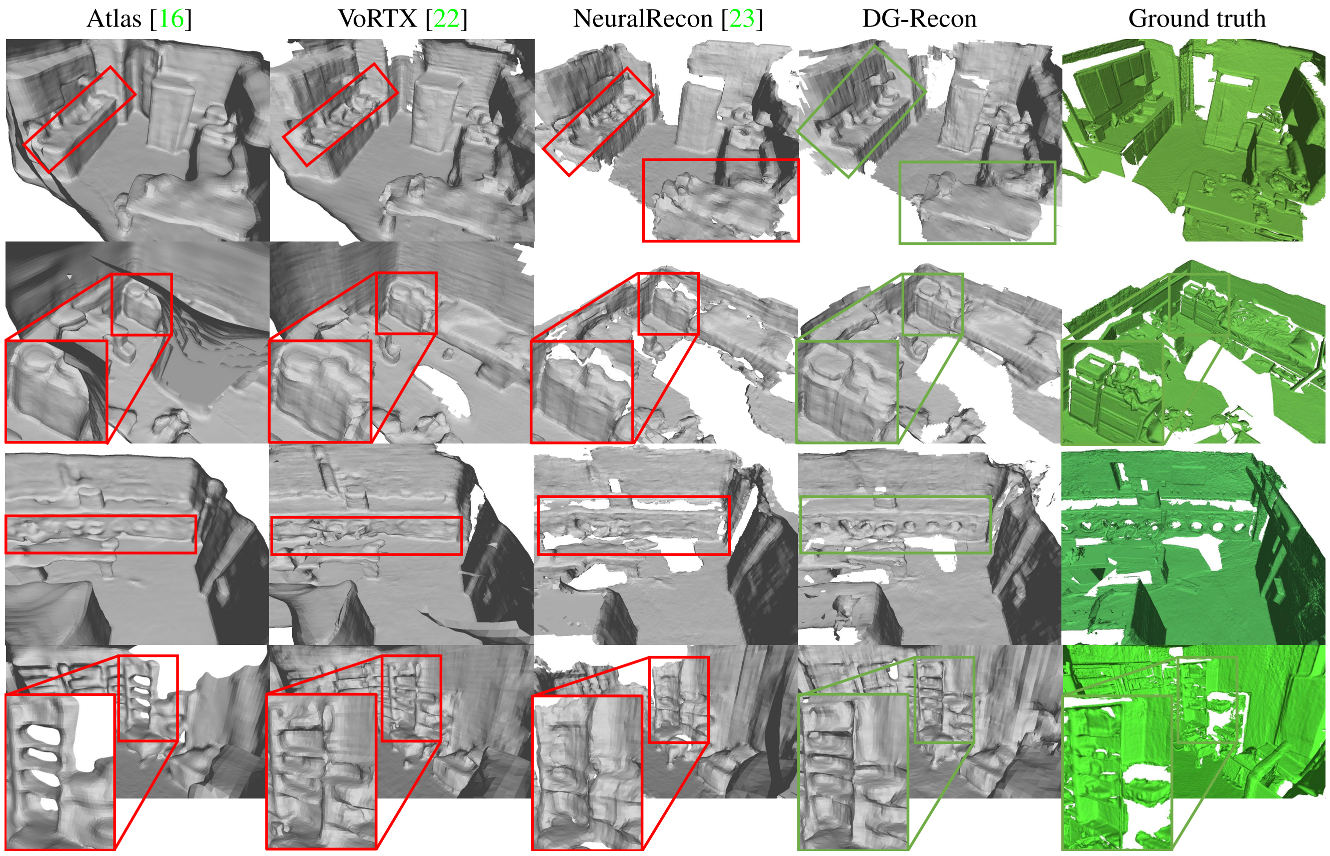


Figure 5. **Qualitative analysis of the volumetric 3D reconstruction methods on the test set of ScanNetV2**. Reconstruction errors are highlighted by red boxes and quality shapes are marked by green boxes. Zoom-in views are provided when necessary.

con [23]. DG-Recon consistently produces more complete reconstruction than the NeuralRecon [23] baseline. Atlas [16] outputs smoother reconstruction for the hotel room (row 1) but completely fails to reconstruct the office room (row 2). Moreover, the third row shows the scale of an entire floor which doesn't fit into the memory of a single NVIDIA 2080Ti GPU for Atlas. Its reconstruction, therefore, misses half of the floor. Note that the failure in the second row is from the non-missing part of the Atlas reconstruction.

5.3. More qualitative analysis on ScanNet

Figure 5 shows more qualitative comparisons for the volumetric reconstruction methods. The worktop (row 1), cabinet and bed (row 2) appear sharper than the other methods. Objects like washing machine (row 3), shelves (row 4) are either incomplete [23] or distorted [16, 22] while ours show a better trade-off between completeness and accuracy.

5.4. Qualitative analysis for outdoor scenes

Even though DG-Recon suffers from distribution shift when generalizing to outdoor scenes like other data-driven methods, it has potential to partially mitigate this shift thanks to its modular design *i.e*. separated depth prior and TSDF prediction. Figure 6 qualitatively shows notable improvement of DG-Recon over Open3D TSDF integration for an outdoor scene given an off-the-shelf monocular depth model, Omnidata [5].

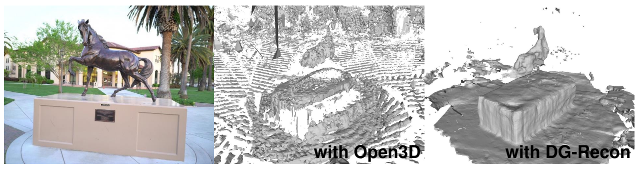


Figure 6. **Outdoor reconstruction example: Horse [12].** Note that relative depths from Omnidata were scaled with COLMAP sparse depths to obtain metric depths.

5.5. Qualitative analysis of rendered depth

We also provide the example depth rendering from NeuralRecon [23] and DG-Recon in Figure 7. The rendered depth of NeuralRecon [23] contains more missing values which might negatively influence the user experience when rendered virtual objects interact with the scene. DG-Recon suffers less from missing depth values and outputs more

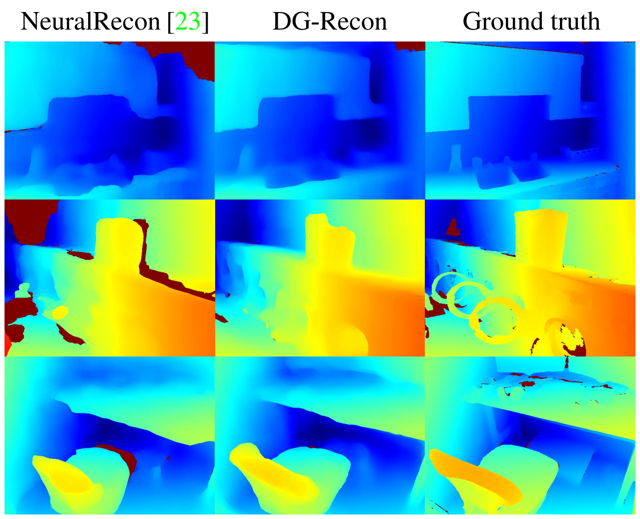


Figure 7. **Qualitative analysis of rendered depth on ScanNetV2** [2]. The color indicates the relative depth ranging from 0 to 1, varying from orange to blue. Dark red indicates missing depth values.

complete object shapes, *e.g*., the chair in row 3.

References

[1] Aljaz Bozic, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. *NeurIPS*, 2021.

[2] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[3] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM TOG*, 2017.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[5] Eftekhar et al. Omnidata. In *ICCV*, 2021.

[6] Ziyue Feng, Leon Yang, Pengsheng Guo, and Bing Li. Cvrecon: Rethinking 3d geometric feature learning for neural reconstruction. 2023.

[7] Huiyu Gao, Wei Mao, and Miaomiao Liu. Visfusion. In *CVPR*, 2023.

[8] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time rgb-d camera relocalization. In *ISMAR*, 2013.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv*, 2014.

[12] Knapitsch et al. Tanks and temples. *TOG*, 2017.

[13] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*. 2002.

[14] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH*, 1987.

[15] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *ICRA*, 1985.

[16] Zak Murez, Tarrence Van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*, 2020.

[17] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

[19] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[20] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[21] Noah Stier, Anurag Ranjan, Alex Colburn, Yajie Yan, Liang Yang, Fangchang Ma, and Baptiste Angles. Finerecon: Depth-aware feed-forward network for detailed 3d reconstruction. 2023.

[22] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer. Vortx: Volumetric 3d reconstruction with transformers for voxelwise view selection and fusion. In *3DV*, 2021.

[23] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *CVPR*, 2021.

[24] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

[25] Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[27] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013.

[28] Xingxing Zuo, Nan Yang, Nathaniel Merrill, Binbin Xu, and Stefan Leutenegger. Incremental dense reconstruction from monocular video with guided sparse feature fusion. *RA-L*, 2023.